DETAILED ACTION

1. This Office Action is response to Applicants' AMENDMENT filed on 03/27/2008.

2. Claims 1-39 were cancelled.

3. Claim 74 has been added

4. Claims 40-74 are pending in this Application.

5. Claims 52-59, 64 and 69 were allowed.

Response to Arguments

6. Applicant's arguments filed 03/27/2008 have been fully considered but they are not persuasive.

101 Rejection (pages 14-15, in the remarks):

The "processing system" recited in the system claims does not overcome the 101 issue. First of all, the "processing system" is lacking the requisite support for the claimed subject matter (claimed subject matter is only presented in the claims)

So, it is rejected with: **Claimed Subject Matter Not in Specification** (The specification is objected to as failing to provide proper antecedent basis for the claimed subject matter. See 37 CFR 1.75(d)(1) and MPEP § 608.01(o). Correction of the following is required: "processing system" is lacking the requisite support for the claimed subject matter (claimed subject matter is only presented in the claims and not in the specification). Applicants are advised to amend the claim(s) that one skilled in the art can make and use the claimed invention). Next, the "system" in the claims of 60 and 64 are lacking a piece of physical object to implement the steps or functions in the claimed

invention. So, it might be a software system (see instant specification, page 7, lines 5-7), software per se. Thus, software per se is non-statutory subject matter.

Applicant argued that, "Caron-Colby-Shan fails to disclose, teach or suggest "stopping the recursive query of ….in the dependency information tracking array."" (pages 16-34, in the remarks).

In response to Appellants' arguments, Examiner respectfully disagrees as Caron teaches the dependency data or information is to record in the table for tracking the dependency information (col. 9, lines 65-67, col. 10, lines 1-10; also col. 7, lines, 37-48 and col. 8, lines 38-52, see figs 11-13) and SHAN teaches the stopping recursion by freeing with the operator to pose the queries (page 16, lines 121 and fig. 3).

Applicant argued that, "Caron-Colby-Shan fails to disclose, teach or suggest "recursively query a database … in the database."" (pages 17-34, in the remarks).

In response to Appellants' arguments, Examiner respectfully disagrees as SHAN teaches recursive query with SQL statements to get the requested information (page 1, the last paragraph, and pages 9-12 under recursive queries section; also, the title of reference clearly said that. Also, see pages 2-7 and 8-11)

Applicant argued that, "Caron-Colby-Shan fails to disclose, teach or suggest "for each of the one or more dependencies … and terminating the recursively query … in the graph." In the claim 72." (pages 19-34, in the remarks).

In response to Appellants' arguments, Examiner respectfully disagrees as Laursen teaches directed graph and tracking arrays (col. 12, lines 35-36, lines 40-45, 52-67 and col. 13, lines 17-25). The claim 72 is a newly converted to independent claim. It is consisting all limitations of claim 40 except the graph, which is come from the old claim of 72.

EXAMINER'S AMENDMENT

7. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Jenni R. Moen (Reg. No.: 52,038) on MON. 06/09/2008 (2:50PM).

The application has been amended as follows:

First line of claim 49, "The method of claim 75," is replaced with "The method of claim 48,"

Claim Rejections - 35 USC § 101

8. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 60-64 rejected under 35 U.S.C. 101 because the system is not specify or missing a piece of physical object to implement the steps or functions in the claimed invention. It is software per se. (see instant specification, page 7, lines 5-7). "Software per se" is non-statutory subject matter under 35 USC 101.

"Software per se" is non-statutory under 35 USC 101 because it is merely a set of instructions without any defined tangible output or tangible result being produced. The requirement for tangible result under 35 USC 101 is defined in *State Street Bank & Trust Co. v. Signature Financial Group Inc.*, 149 F.3d 1368, 47USPQ2d 1596 (Fed. Cir. 1998).

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein

were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

Claims 40-42, 46-51, 60-62, 65-67 and 73-74 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No.: US 5,586,328 issued to Caron et al. (hereinafter Caron) in view of US Patent No.: US 6,199,063 B1 issued to Colby et al. (hereinafter Colby), and further in view of WO 92/15066 (PCT/US92/01458) issued to Ming-Chien SHAN dated 02/26/1991 (hereinafter SHAN).

With respect to claim 40, Caron a method of generating dependency information for code objects stored in a database (figs. 4 and 8, col. 10, lines 25-32); comprising:

dependencies of procedural code objects stored in the database (the layout dependency code is stored on the storage device or in main memory item 38 in fig. 1: col. 6, lines 30-38 and col. 4, lines 48-50); and

generating a dependency information tracking array based on the indication of one or more dependencies of procedural code objects (figs. 4 and 8, creating a code dependency by a procedure call: col. 10, lines 25-32); and

dependency information tracking array (the dependency data or information is to record in the table for tracking the dependency information (col. 9, lines 65-67, col. 10, lines 1-10; also col. 7, lines, 37-48 and col. 8, lines 38-52, see figs 11-13).

Caron teaches generating dependency code layout for the object code from a program storing on the main memory. Caron further teaches recursive method for generating dependency code (see fig. 12 and col. 15, lines 40-45). Caron does not explicitly teach querying a database and identifying code objects stored in the database as claimed.

However, Colby teaches using SQL statement or command to query a relational database (fig. 6, col. 3, lines 10-25, col. 7, lines 60-67 and col. 8, lines 1-35); identifying the database objects (col. 8, lines 5-35 and col. 15, lines 30-50).

Therefore, based on Caron in view of Colby, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified the teachings of Caron with the teachings of Colby to utilize the use of SQL statement to query on the database and identifying the objects in the database as disclosed (Colby's figs. 6, and col. 8, lines 1-35), into the system of Caron for the purpose of implementing queuing and recursion the data stored in a database, thereby, enabling a user to have a quickly and efficiently deriving answer to a relational database query (Colby's col. 1, lines 20-22 and col. 3, lines 65-67). In combination of Caron and Colby do not teach explicitly recursively querying a database using a SQL statement and stopping the recursive query of the database upon identifying a dependency as claimed.

However, SHAN teach recursive query with SQL statements to get the requested information (page 1, the last paragraph, and pages 9-12 under recursive queries section) and stopping the recursion by freeing with the operator to pose the queries (page 16, lines 121 and fig. 3).

Therefore, based on Caron in view of Colby, and further in view of SHAN, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to modify the teachings of SHAN to the system of Caron to have recursively a SQL statement to query a database with objects stored in the relational database with codes stored in the memory. One having ordinary skill in the art would have found it motivated to utilize the use of recursive query with SQL statement on a database to obtain the requested information from a database as disclosed (SHAN's pages 9-12), into the system of Caron for the purpose of evaluating a recursive query of a database (SHAN's page 1), thereby, enabling the user to translate a query in a form for efficient evaluation in large and complex database systems (SHAN's page 11).

With respect to claim 41-42, Caron a method of generating dependency information for code objects stored in a database (figs. 4 and 8, col. 10, lines 25-32); comprising:

dependencies of procedural code objects stored in the database (the layout dependency code is stored on the storage device or in main memory item 38 in fig. 1: col. 6, lines 30-38 and col. 4, lines 48-50); and

generating a dependency information tracking array based on the indication of one or more dependencies of procedural code objects and incorporating the one or more dependencies of specifications of object-oriented code objects into the dependency information tracking array (figs. 4 and 8, creating a code dependency by a procedure call: col. 10, lines 25-32).

Caron teaches generating dependency code layout for the object code from a program storing on the main memory. Caron further teaches recursive method for generating dependency code (see fig. 12 and col. 15, lines 40-45). Caron does not explicitly teach querying a database and identifying code objects stored in the database.

However, Colby teaches using SQL statement or command to query a relational database (fig. 6, col. 3, lines 10-25, col. 7, lines 60-67 and col. 8, lines 1-35); identifying the database objects (col. 8, lines 5-35 and col. 15, lines 30-50).

Therefore, based on Caron in view of Colby, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified the teachings of Caron with the teachings of Colby to utilize the use of SQL statement to query on the database and identifying the objects in the database as disclosed (Colby's figs. 6, and col. 8, lines 1-35), into the system of Caron for the purpose of implementing queuing and recursion the data stored in a database, thereby, enabling a user to have a quickly and efficiently deriving answer to a relational database query (Colby's col. 1, lines 20-22 and col. 3, lines 65-67). In combination of Caron and Colby do not teach explicitly recursively querying a database using a SQL statement.

However, SHAN teach recursive query with SQL statements to get the requested information (page 1, the last paragraph, and pages 9-12 under recursive queries section).

Therefore, based on Caron in view of Colby, and further in view of SHAN, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to modify the teachings of SHAN to the system of Caron to have recursively

a SQL statement to query a database with objects stored in the relational database with codes stored in the memory. One having ordinary skill in the art would have found it motivated to utilize the use of recursive query with SQL statement on a database to obtain the requested information from a database as disclosed (SHAN's pages 9-12), into the system of Caron for the purpose of evaluating a recursive query of a database (SHAN's page 1), thereby, enabling the user to translate a query in a form for efficient evaluation in large and complex database systems (SHAN's page 11).

With respect to claim 46, Caron teaches further comprising compiling one or more code objects stored in the database in debug mode using a database code object debugging tool (processing of debugging source code to a compiled program: col. 1, lines 55-60).

With respect to claim 47, Caron in view of Colby teaches further comprising identifying one or more dependent objects stored in the database that are INVALID (Colby's fig. 19, col. 21, lines 1-18).

With respect to claim 48, Caron a method of generating dependency information for code objects stored in a database (figs. 4 and 8, col. 10, lines 25-32); comprising:

dependencies of procedural code objects stored in the database (the layout dependency code is stored on the storage device or in main memory item 38 in fig. 1: col. 6, lines 30-38 and col. 4, lines 48-50); and

generating a dependency information tracking array based on the indication of one or more dependencies of procedural code objects (figs. 4 and 8, creating a code dependency by a procedure call: col. 10, lines 25-32); and

dependency information tracking array (the dependency data or information is to record in the table for tracking the dependency information (col. 9, lines 65-67, col. 10, lines 1-10; also col. 7, lines, 37-48 and col. 8, lines 38-52, see figs 11-13).

Caron teaches generating dependency code layout for the object code from a program storing on the main memory. Caron further teaches recursive method for generating dependency code (see fig. 12 and col. 15, lines 40-45). Caron does not explicitly teach querying a database and identifying code objects stored in the database and identifying one or more cyclic dependencies among code objects stored in the database as claimed.

However, Colby teaches using SQL statement or command to query a relational database (fig. 6, col. 3, lines 10-25, col. 7, lines 60-67 and col. 8, lines 1-35); identifying the database objects (col. 8, lines 5-35 and col. 15, lines 30-50; and col. 7, lines 4-15 and col. 10, lines 18-32).

Therefore, based on Caron in view of Colby, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified the teachings of Caron with the teachings of Colby to utilize the use of SQL statement to query on the database and identifying the objects in the database as disclosed (Colby's figs. 6, and col. 8, lines 1-35), into the system of Caron for the purpose of implementing queuing and recursion the data stored in a database, thereby, enabling a user to have a quickly and efficiently deriving answer to a relational database query (Colby's col. 1, lines 20-22 and col. 3, lines 65-67). In combination of Caron and Colby do not teach explicitly recursively querying a database using a SQL statement as claimed.

However, SHAN teach recursive query with SQL statements to get the requested information (page 1, the last paragraph, and pages 9-12 under recursive queries section).

Therefore, based on Caron in view of Colby, and further in view of SHAN, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to modify the teachings of SHAN to the system of Caron to have recursively a SQL statement to query a database with objects stored in the relational database with codes stored in the memory. One having ordinary skill in the art would have found it motivated to utilize the use of recursive query with SQL statement on a database to obtain the requested information from a database as disclosed (SHAN's pages 9-12), into the system of Caron for the purpose of evaluating a recursive query of a database (SHAN's page 1), thereby, enabling the user to translate a query in a form for efficient evaluation in large and complex database systems (SHAN's page 11).

With respect to claim 49, Colby teaches wherein identifying one or more cyclic dependencies comprises utilizing a graph traversal algorithm to identify one or more cyclic dependencies (col. 7, lines 4-15 and col. 10, lines 18-32).

With respect to claim 50, Colby teaches further comprising generating a dependency graph for code objects stored in the database based at least in part on the dependency information tracking array (col. 7, lines 4-15 and col. 10, lines 18-32).

Claim 60 is essentially the same as claim 40 except that it is directed to a system rather than a method, and is rejected for the same reason as applied to the claim 40 hereinabove.

Claim 61 is essentially the same as claim 41 except that it is directed to a system rather than a method, and is rejected for the same reason as applied to the claim 41 hereinabove.

Claim 62 is essentially the same as claim 42 except that it is directed to a system rather than a method, and is rejected for the same reason as applied to the claim 42 hereinabove.

Claim 65 is essentially the same as claim 40 except that it is directed to a computer readable medium rather than a method, and is rejected for the same reason as applied to the claim 40 hereinabove.

Claim 66 is essentially the same as claim 41 except that it is directed to a computer readable medium rather than a method, and is rejected for the same reason as applied to the claim 41 hereinabove.

Claim 67 is essentially the same as claim 42 except that it is directed to a computer readable medium rather than a method, and is rejected for the same reason as applied to the claim 42 hereinabove.

10. Claims 43-45, 51, 63 and 68 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No.: US 5,586,328 issued to Caron et al. (hereinafter Caron) in view of US Patent No.: US 6,199,063 B1 issued to Colby et al. (hereinafter Colby) and further in view of WO 92/15066 (PCT/US92/01458) issued to Ming-Chien SHAN dated 02/26/1991 (hereinafter SHAN) and in view of US Patent No. 5,926,819 issued to Doo et al. (hereinafter Doo).

With respect to claim 43, Caron in view of Colby and SHAN discloses a method of generating dependency information as discussed in claim 40.

Caron, Colby and SHAN disclose substantially the invention as claimed.

Caron, Colby and SHAN do not teach parsing the source code of the database for data manipulation statements that fire triggers; and identifying one or more data manipulation statements that fire triggers.

However, Doo teaches DML statement being applied to fire the triggers (col. 5, lines 22-42, also see figs, 2-4) and parsing the tree of DML statement (col. 5, lines 38-42 and col. 6, lines 50-58).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made for modifying the teachings of Caron in view of Colby and SHAN with the teachings of Doo by incorporating the use of data manipulation statements such as DML to fire the triggers. The motivation being for having secure triggers in order to prevent the replication or even corrupt other data in the database (Doo's col. 2, lines 58-62).

With respect to claim 44, Caron in view of Colby and SHAN discloses a method of generating dependency information as discussed in claim 40.

Caron, Colby and SHAN disclose substantially the invention as claimed.

Caron, Colby and SHAN do not teach parsing the source code for UPDATE, DELETE or INSERT statements.

However, Doo teaches DML statements being applied to INSERT, DELETE and UPDATE operations (col. 5, lines 5-12).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made for modifying the teachings of Caron in view of Colby and SHAN with the teachings of Doo by incorporating the use of data manipulation statements such as DML to fire the triggers. The motivation being for having secure triggers in order to prevent the replication or even corrupt other data in the database (Doo's col. 2, lines 58-62).

With respect to claim 45, Caron in view of Colby teaches a method of generating dependency information as discussed in claim 40.

Caron teaches generating dependency code layout for the object code from a program storing on the main memory. Caron further teaches recursive method for generating dependency code (see fig. 12 and col. 15, lines 40-45). Colby teaches using SQL statement or command to query a relational database and identifying the database objects.

Therefore, based on Caron in view of Colby, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified

the teachings of Caron with the teachings of Colby to utilize the use of SQL statement to query on the database and identifying the objects in the database as disclosed (Colby's figs. 6, and col. 8, lines 1-35), into the system of Caron for the purpose of implementing queuing and recursion the data stored in a database, thereby, enabling a user to have a quickly and efficiently deriving answer to a relational database query (Colby's col. 1, lines 20-22 and col. 3, lines 65-67). In combination of Caron and Colby do not teach explicitly recursively querying a database using a SQL statement.

However, SHAN teach recursive query with SQL statements to get the requested information (page 1, the last paragraph, and pages 9-12 under recursive queries section).

Therefore, based on Caron in view of Colby, and further in view of SHAN, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to modify the teachings of SHAN to the system of Caron to have recursively a SQL statement to query a database with objects stored in the relational database with codes stored in the memory. One having ordinary skill in the art would have found it motivated to utilize the use of recursive query with SQL statement on a database to obtain the requested information from a database as disclosed (SHAN's pages 9-12), into the system of Caron for the purpose of evaluating a recursive query of a database (SHAN's page 1), thereby, enabling the user to translate a query in a form for efficient evaluation in large and complex database systems (SHAN's page 11).

With respect to claim 51, Caron in view of Colby and SHAN discloses a method of generating dependency information as discussed in claim 40.

Caron, Colby and SHAN disclose substantially the invention as claimed.

Caron, Colby and SHAN do not teach database catalog.

However, Doo teaches the database system stores information as metadata in a data dictionary within the database (col. 6, lines 27-32).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made for modifying the teachings of Caron in view of Colby and SHAN with the teachings of Doo by incorporating the use of data manipulation statements such as DML to fire the triggers. The motivation being for having secure triggers in order to prevent the replication or even corrupt other data in the database (Doo's col. 2, lines 58-62).

Claim 63 is essentially the same as claim 43 except that it is directed to a system rather than a method, and is rejected for the same reason as applied to the claim 43 hereinabove.

Claim 68 is essentially the same as claim 43 except that it is directed to a computer-readable medium rather than a method, and is rejected for the same reason as applied to the claim 43 hereinabove.

11. Claims 70-74 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No.: US 5,586,328 issued to Caron et al. (hereinafter Caron) in view of US Patent No.: US 6,199,063 B1 issued to Colby et al. (hereinafter Colby) and further in view of WO 92/15066 (PCT/US92/01458) issued to Ming-Chien SHAN dated 02/26/1991 (hereinafter SHAN) and in view of US Patent No. 5,805,804 issued to Laursen et al. (hereinafter Laursen).

With respect to claims 70-71, Caron in view of Colby and SHAN discloses a method of generating dependency information as discussed in claim 40.

Caron, Colby and SHAN disclose substantially the invention as claimed. Caron, Colby and SHAN do not teach PL/SQL specifications for a collection of stored functions and procedures identified as a single entity.

However, Laursen teaches PL/SQL statements (detx 59)

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made for modifying the teachings of Caron in view of Colby and SHAN with the teachings of Laursen by incorporating the use of PL/SQL for collecting of stored functions. The motivation being for selecting, collecting, retrieving and delivering information over the network in real-time or non-real-time (Laursen's col. 1, lines 10-10 and col. 2, lines 1-12).

With respect to claim 72, Caron a method of generating dependency information for code objects stored in a database (figs. 4 and 8, col. 10, lines 25-32); comprising:

dependencies of procedural code objects stored in the database (the layout dependency code is stored on the storage device or in main memory item 38 in fig. 1: col. 6, lines 30-38 and col. 4, lines 48-50); and

generating a dependency information tracking array based on the indication of one or more dependencies of procedural code objects (figs. 4 and 8, creating a code dependency by a procedure call: col. 10, lines 25-32); and

dependency information tracking array (the dependency data or information is to record in the table for tracking the dependency information (col. 9, lines 65-67, col. 10, lines 1-10; also col. 7, lines, 37-48 and col. 8, lines 38-52, see figs 11-13).

Caron teaches generating dependency code layout for the object code from a program storing on the main memory. Caron further teaches recursive method for generating dependency code (see fig. 12 and col. 15, lines 40-45). Caron does not explicitly teach querying a database and identifying code objects stored in the database as claimed.

However, Colby teaches using SQL statement or command to query a relational database (fig. 6, col. 3, lines 10-25, col. 7, lines 60-67 and col. 8, lines 1-35); identifying the database objects (col. 8, lines 5-35 and col. 15, lines 30-50).

Therefore, based on Caron in view of Colby, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified the teachings of Caron with the teachings of Colby to utilize the use of SQL statement to query on the database and identifying the objects in the database as disclosed (Colby's figs. 6, and col. 8, lines 1-35), into the system of Caron for the purpose of implementing

queuing and recursion the data stored in a database, thereby, enabling a user to have a quickly and efficiently deriving answer to a relational database query (Colby's col. 1, lines 20-22 and col. 3, lines 65-67). In combination of Caron and Colby do not teach explicitly recursively querying a database using a SQL statement and stopping the recursive query of the database upon identifying a dependency as claimed.

However, SHAN teach recursive query with SQL statements to get the requested information (page 1, the last paragraph, and pages 9-12 under recursive queries section) and stopping the recursion by freeing with the operator to pose the queries (page 16, lines 121 and fig. 3).

Therefore, based on Caron in view of Colby and SHAN, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have modified the teachings of Caron with the teachings of Colby and SHAN. One having ordinary skill in the art would have found it motivated to utilize the use of recursive query with SQL statement on a database to obtain the requested information from a database as disclosed (SHAN's pages 9-12), into the system of CARON. In combination of Caron and Colby and SHAN do not teach explicitly one of the one or more dependencies already occurs in the graph and displaying a dependency graph.

However, Laursen teaches directed graph and tracking arrays (col. 12, lines 35-36, lines 40-45, 52-67 and col. 13, lines 17-25).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made for modifying the teachings of Caron in view of Colby and SHAN with the teachings of Laursen by incorporating the use of PL/SQL for

collecting of stored functions. The motivation being for selecting, collecting, retrieving and delivering information over the network in real-time or non-real-time (Laursen's col. 1, lines 10-10 and col. 2, lines 1-12).

With respect to claims 73-74, Caron in view of Colby and SHAN discloses a method of generating dependency information as discussed in claim 40.

Caron, Colby and SHAN disclose substantially the invention as claimed.

Caron, Colby and SHAN do not teach to a user, the dependency graph generated based at least in part on the dependency information tracking array..

However, Laursen teaches directed graph and tracking arrays (col. 12, lines 35-36, lines 40-45, 52-67 and col. 13, lines 17-25).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made for modifying the teachings of Caron in view of Colby and SHAN with the teachings of Laursen by incorporating the use of PL/SQL for collecting of stored functions. The motivation being for selecting, collecting, retrieving and delivering information over the network in real-time or non-real-time (Laursen's col. 1, lines 10-10 and col. 2, lines 1-12).

Allowable Subject Matter

12. The following is a statement of reasons for the indication of allowable subject matter: Claims 52-59, 64 and 69 are allowed because these claims have some distinct features: "parsing the source code of the database for data manipulation statements that fire triggers and generate a dependency information tracking array based on the indications of one or more dependencies of procedural code objects stored in the database, one or more dependencies of specifications of object-oriented code objects stored in the database, one or more dependencies of implementations of object-oriented code objects stored in the database and one or more dependencies on triggers of code objects stored in the database." These distinct features, in conjunction with all other limitations of the dependents and independent claims render claims 52-59, 64 and 69 them allowable. If they are overcome the 101 rejection above.

13. Claims 52-59, 64 and 69 are allowed.

Conclusion

14. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Contact Information

15. Any inquiry concerning this communication or earlier communications from the examiner should be directed to ANH LY whose telephone number is (571) 272-4039 or via E-Mail: ANH.LY@USPTO.GOV (Written Authorization being given by Applicant (MPEP 502.03 [R-2])) or fax to **(571) 273-4039** (unofficial fax number directly to examiner's office). The examiner can normally be reached on TUESDAY – THURSDAY from 8:30 AM – 3:30 PM. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, **John Breene**, can be reached on **(571) 272-4107** or Primary Examiner, **Jean Fleurantin**, can be reached on **(571) 272-4035**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). Any response to this action should be mailed to: Commissioner of Patents and Trademarks, Washington, D.C. 20231, or faxed to:

Central Fax Center: (571) 273-8300.

ANH LY /AL/
JUN. 6th, 2008

/JEAN B. FLEURANTIN/
Primary Examiner, Art Unit 2162